NaPP: Notification and Push Performance in Wearable Devices

Taniza Sultana and Kristen R. Walcott

University of Colorado Colorado Springs
tsultana@uccs.edu kwalcott@uccs.edu

Abstract. The Internet of Things (IoT) and other related systems have dramatically changed how we interact with the world. Our devices provide us with "immediate" information to our connected smart wearable devices, smartphones, and other tools. We rely on receiving notifications regardless of the proximity of the device. Notification response is especially important for medical alerts and urgent calls. Notification delays occur for many reasons ranging from networks, hardware, or the applications themselves. Given the importance of on-time notifications, a better understanding of notification delays is needed.

In this work, we study notification delays in Android mobile/wearable devices and examine trends in notification delays across devices. We analyze thirty-two devices based on notification pushes, and we observe that there are delays between the smartphone generating notification pushes and the connected wearable device receiving said notification. The delay is especially observed from notification push performance on older devices. We then identify several vectors of hardware and network aspects that cause delays.

Keywords: Wearable devices, smartwatch, smartphone notifications, delay tolerant network, push notifications, flash memory

1 Introduction

The popularity of wearable devices has grown significantly over the past few years. It is estimated that the wearables global market will reach $32.9 billion by 2020 [2]. These include devices such as smart watches and others. Network connectivity is vital for these devices since most of the wearable devices are considered low resource devices due to their physical and memory size [19].

One function that is particularly important in the use of these devices is reliable comminication and notification. This includes the receiving and displaying of text and call notifications from connected smartphones on the wearable or other connected devices. These appear as *"smartphone notifications"* or *"push notifications"* [9].

Wearable devices and others have notifications/alert functions. These hardware notifications allow the device to receive incoming text/call alerts from their con-

nected smartphone while the device is not in close proximity but within the maximum Bluetooth (BT) range of the connected wearable device. Given that these devices have their limitations within the BT connectivity range and that physical proximity plays a role in whether it will receive smartphone notifications or not, a user would hope that the delay of receipt of a notification would be small.

Unfortunately, the performance of a notification push can be critical. If notifications are arriving at different times on different devices, it can be problematic. Delays in receiving a notification may defeat the purpose of the notifications alert function in wearable devices. Users prefer this function so that they can continue receiving messages and call alerts while not in close proximity to their smartphone. Call alerts often are more critical compared to other messaging alerts, especially if it is a time sensitive call (i.e. an emergency call requiring immediate attention). Receiving delayed call alerts while away from the smartphone will affect users in that they are then delayed from reaching their phone to answer or respond.

Wearable devices are also critical for health monitoring and tracking. These devices use sensor technology and notifications to assist in moitoring [22]. For example, medical alert notifications of an abnormal condition or pre-set alert trigger allow a medical device to send alerts to its user or data receiver. From someone such as a healthcare provider may review the data to detect an abnormality [13]. Wearable technology such as HealthPatch MD and Xio XT can even generate alerts for healthcare providers. If there is a delay in notifications/alert process in wearable devices, medical alert notifications are likely to be affected as well [14].

Given the many devices that are used over many networks in different environments, it is important for the users to be well informed if their devices have potential notification delays in certain conditions or configurations. This can include device types, models, networks and environments. It is essential to identify all factors that can affect a smartphone's push notifications that can increase latency in notification alerts to associated wearable devices.

To observe and evaluate the notification push performance (NPP) between smartphones and connected wearable devices, we test thirty two Android phones and their connected smartwatches. Push performance are measured through four different configurations where the call notification push, call and text notification push, push during wifi vs network connection, and push after completing updates on phone were observed. From this, we identify several vectors that affect the push notification performance such as excessive stored data, older model device, and phone carrier service.

The main contributions of this paper are as follows:

– Description of an experiment on NaPP across 32 devices and 4 configurations (Section 3)

– Analysis of NaPP data (Section 4)

– Discussion of a vector of mobile properties affecting NaPP (Section 5)

2 Notification Processes and Paired Wearable Communication Methods

The smartphone push notification process starts with the device trying to gain user's attention through their paired wearable device and its preferred settings (i.e. vibrations, sounds or visual) [20]. With a real-time push notifications the smartphone system continuously monitors streams to alert the users. Real-time push notifications usually alert the users immediately after a notification is generated in their mobile device [16]. However for their connected wearable devices, the push notifications are often not performed at the same speed even if the devices follow the same communication protocols. For example, a push notification may take several seconds before waking up the connected wearable to perform the push notification. Often this is due to the device's embedded delay-tolerant network (DTN) system. Bluetooth communication also has an impact on notification timing on many wearable devices.

2.1 Paired Wearable Communication

One-way and two-way communications can occur between smartphones and their paired devices. In a one-way communication, as shown in Figure 1, when a smartphone receives an alert such as call or text, it alerts user app on phone side for its paired devices to wake up the notification manager. The notification manager posts the notification to a the other devices service (for example, an Android wear service) to validate settings and preferences. Upon approval, it then pushes the notification to the associated device service. Once the receiving service gets the post, an alert is shown on the device, and the smartphone's notification push for one-way communication process is complete [9].

Two-way communication is performed when a device is linked with a corresponding manufacturers device application on the paired smartphone. This allows the two devices to be able to communicate and exchange data. As shown as an example in Figure 2, for the mobile device to push a notification to the wearable, it first recieves an alert, then it wakes up the wearable device and pushes the notification. The wearable device will then validate and compare to the user settings (Do Not Disturb, nighttime mode, etc) and it will provide the notification. In turn, the wearable device and push data to the paired mobile device application. when the watch-side user app takes action and communicates back to the Android device through the "wear service". In this case, the watch-side app service will determine the push response and any return actions back to phone app service [9].

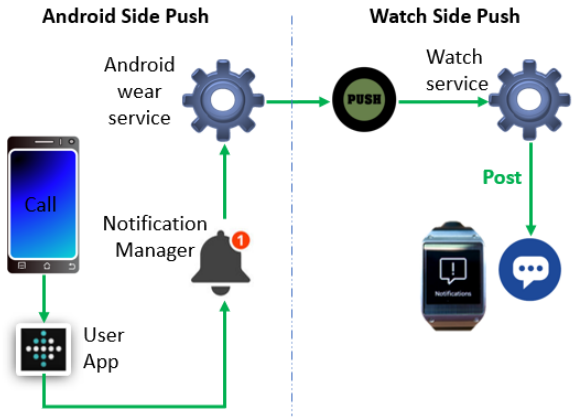

Fig. 1: One-way communication between smartphone and wearable/smartwatch (adapted from [9])

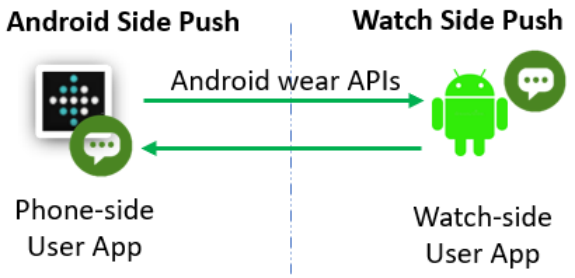

Fig. 2: Two-way communication between smartphone and wearable/smartwatch (adapted from [9])

2.2 Bluetooth Communication

To save energy and battery power, many IoT and wearable devices rely on "low-power wireless communication" such as Bluetooth Low Energy (BLE). Almost all smartwatches use BLE communication [23]. BLE communication links one wearable device to only one smartphone which causes the smartwatch to lose its ability to receive notifications while the linked smartphone is either turned off or outside of its connectivity range [23].

Wearable connected devices mostly communicate via BT, BLE or built-in Wi-fi [9] [10]. In some cases, the Bluetooth range (BT-range) is intentionally kept short to save energy. As a result, many devices frequently handover between BT and Wi-fi repetitively. With Android devices, the handover can take long time to finish, causing a delay in their push notifications [10]. When both devices are connected to Wi-fi, they gnerally communicate through their designated servers (i.e. Google's servers).

3 Experimental Approach and Evaluation

Embedded system testing with connected mobile devices is a complex process, and test results may vary between configurations. We conducted a comparison study by manually testing thirty two sets of devices (various smartwatches and connected Android phones) in four different configurations. We collected and analyzed the data while leveraging comparative studies, which assisted in determining the following research objectives:

- Examined what, if any, correlation exists between a smartwatch call notification delay and the physical proximity between the smartwatch and connected smartphone.
- Analyzed the various delays between the different types of smartwatches and the various types of connected smartphones.
- Analyzed the delay between a smartphone's Wi-fi connection and phone carrier service.
- Determined if there is a significant difference between devices with the least-delay and devices with the most-delay in notification alerts.

Through manual testing, calls were made to the smart phones to record call notification times. The delay is then measured in seconds. We recorded the time between the moment a call displayed on the phone to the moment call notification displayed on the smartwatch. Each set was tested multiple times with the phone and smartwatch at different distances and different configurations. In addition, during the study, each phone's available memory and service carrier's information were collected.

4 NaPP Data Evaluation and Analysis

Based on our results from four configurations described below, we compare devices, analyze data from all low performing devices and evaluate factors that contribute to push notification delays.

4.1 Device Comparison

The two sets of devices from our initial observations were compared side by side to determine separating factors between them. In our initial observation, the first device (C1) performed poorly compared to the second device (C2). We evaluated both sets of devices and swapped their connected smartwatches to see if their performance changed based on their connected device. There were no notable changes in the performance during the swap. Aside from their design space and network, we also evaluated each of the devices settings, available memory space, and application updates as seen in Table 1.

Although the Android phones under test are the same model and on the same cellular network, device state differed. Two factors that stand out, as seen in

Evaluation Matrix	**Samsung Galaxy S7 (C1)**	**Samsung Galaxy S7 (C2)**
Wi-fi Network	✓Same	✓Same
Bluetooth Settings	✓Same	✓Same
Carrier Service	Sprint	Sprint
Test Location	✓Same	✓Same
Watch App Settings	✓Same	✓Same
Device Memory	32GB	32GB
Stored Data Volume	31.2GB	30GB
Free Memory	856MB	2GB
Android Service Setting	Do Not Disturb	None
Application Updates	None	None

Table 1: Android Devices Evaluation & Comparison

Configuration	**Description**
1	Call Notifications On/Off
2	Call and Text Notifications On/Off
3	Change of Carrier Services
4	Outstanding Application Updates

Table 2: Experiment Configurations

Table 1, are the devices' available memory space and Android service settings. The low performing device (C1) had significantly less memory space available compared to the higher performing device (C2). Device C1 also was set to "do not disturb" mode whereas C2 was not. To confirm if these identified factors might be contributing factors in the push notifications delay, we first removed the "do not disturb" setting to allow all calls to come through. We then cleared memory space to increase available memory space for the device (C1). Once both changes were made to the C1 device, we performed the test again under the same four configurations. Given these changes, performance improved by 75% on the C1 device. In analyzing other devices, low memory space(200MB) consistently resulted in slower notification pushes. In the case of a "do not disturb" setting being on, a push will be never performed.

4.2 Notification Push Performance (NaPP) Data Analysis

Following initial device comparison, we next test all devices in four different configurations. In the first configuration, only call notification is on. In the second configuration, both call and text notifications are enabled. Configuration 3 modifies the carrier service, and Configuration 4 examines outstanding application updates, as shown in Table 2.

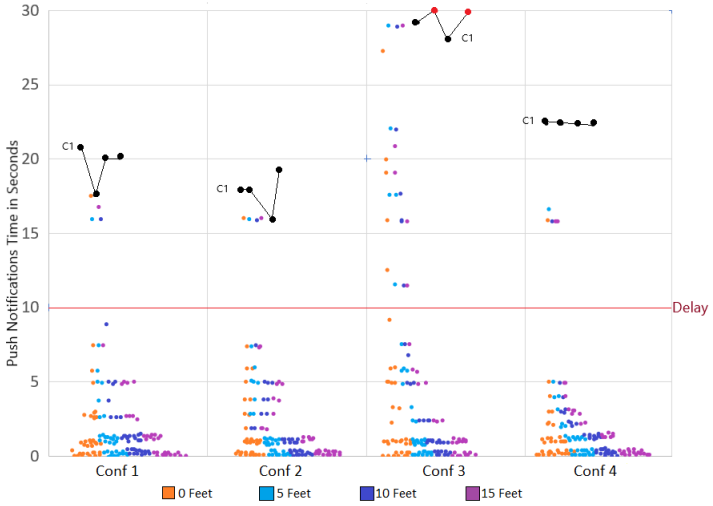

Fig. 3: Configurations and Data Analysis

A comparison of the push notification times of the 32 devices across the 4 configurations can be seen in Figure 3. As in the initial device comparison, all devices are the same model and on the same cellular network. We additionally examine the impact of moving the devices away from the source of notification to determine the impact of Bluetooth range.

In Figure 3, each configuration's data is displayed in individual case study blocks, where y values are the push response time in seconds. Distance between testing devices are color-coded in four categories. A "delay" is defined as a notification that has not been received within 10 seconds. Push notifications that occur within 5 to 10 seconds are considered as "no notable delay." The worst performing device C1, described in Table 1, is displayed in connected lines where a red dot indicates that the device was unable to perform the notification push.

The data from Configuration 1 (only call notification on) and Configuration 2 (call and text notifications on) were compared to determine if reducing push service volume from multiple applications can improve Android wear service push notifications performance. There were no differences observed in notification push performance if one application or multiple applications were enabled for Android wear service to perform the push. The Android wear service processes the call notification as soon an alert is available on phone-side apps and immediately moves it to push determination. It then waits for the next available alert without causing any delay.

We also analyzed the data from Configuration 3 to identify if placing the device under a different phone carrier service has any effect on push notifications. We observed performance dropped (i.e. the notification push time increased) in eight sets of devices compared to their data collected during Configuration 1 and

Configuration 2 observations. Those sets of devices performance returned to their baseline when tested again under their Wi-fi connection. This data indicated that push notification can be affected when phones are performing under a carrier network where low network connectivity can delay the overall push notifications.

In many cases, smartphone users may pause an update for later and forget to return to complete the upgrade. Configuration 4 was designed to capture any unnoticed or hidden application updates in the phones. In our study, we identified two devices that had pending application (image gallery and Venom) updates, which caused notification delays. Once the applications were updated on both devices and tested again, we noted that the notification performance on the devices improved immediately.

4.3 Research Objectives Review

Based on the evaluation and the collected data analysis, we were able to make determinations on the four research objectives we listed earlier in Section 3. **There is no correlation between smartwatch call notification delay and the physical proximity between the smartwatch and connected smartphone.**

Based on all four configurations and analysis, there was no difference noted in the call notification push at four distances ranging from zero to fifteen feet between the smartphones and their associated smartwatches. Therefore, we determined that physical proximity does not play any role in call notification push unless the devices are outside of their BT-range. In that case, the connected smartwatch will not receive any notification at all and rather dropped the push notification if outside of a certain BT-range.

Call notification push does not vary between smartwatch types, but it may vary between smartphone types.

In our work, some devices performed poorly in notification push. For those sets of devices, we looked at both the phone and smartwatch separately to determine if the delay notification behavior is in the phone or in the smartwatch. Those specific smartwatches did not display the same behavior when they were connected with another model smartphone. The delay behavior in call notification push was mostly observed in older models (Samsung Galaxy S6 and Samsung Galaxy S7).

There is a slight difference in call notifications push between Wi-fi connection versus phone carrier service.

In most pairs of devices, there was no notable difference in call notification push between testing under Wi-fi connection and testing under phone carrier service; 24% of the devices between two carrier services displayed latency in their call notification pushes compared to using a Wi-fi connection.

There is no significant difference in hardware design, software and architecture, between low performing and high performing devices other than their memory space, and processing power.

Even though there is not a significant difference in the architecture or software of low performing and high performing devices, the memory space did play a role in the speed of push notification processing. Based on the data collected, all newer smartphones have greater memory available compared to older versions of the phones. Push delay was negligible between Samsung Galaxy 9 and 10 models. These models also have installed memory space between 64MB to 128MB and expanded memory up-to 512MB.

In summary, we observed the notification push performance of newer Android phones is faster than older Android devices and within an acceptable range in our work. We also examined the performance of older devices which are still in use by people around the world for various reasons, especially among the older population. We found the older devices had notable delays with push notifications which could potentially be an issue for wearable and health device manufacturers as many health devices rely on the same concepts for their health alert notifications.

5 Delay Vector Classification

Based on the evaluation data, we conclude that there is no notable correlation between smartwatch call notification push and the physical proximity between the smartwatch and connected smartphone. We also determine that call notification pushes may vary between smartphone types (i.e. older version vs newer phones). We note from our observation that there is a small difference in call notification pushes between Wi-fi connection compared to phone carrier services. In the study, all newer smartphones have greater memory available compared to the older versions of the phone. Push delay is almost non-existent in Samsung Galaxy 9 and 10 models. These models also have installed memory space between 64MB and 128MB with expanded memory up to 512MB.

To create a delay vector, we identified factors with higher probability to disrupt the smartphone's push notifications as listed below:

5.1 Delay Tolerant Network

The routing protocols in Delay Tolerant Networks (DTN) adapt themselves even when they are not continuously connected to their paired device. When in an environment when the device is not connected to Wi-fi, the routing protocol propagates multiple copies of data packets to increase the probability of the delivery [1]. Our study reveals that DTN does not affect push notifications when the devices are on the same Wi-fi network. However, when using a carrier service, DTN processes data in using a multiple packets delivery method and

Type	Conf 1	Conf 2	Conf 3	Conf 4
Galaxy S6	6.2	6.05	15	5.5
Galaxy S7	2.56	3	8.69	2.19
Galaxy S8	2.25	3	2	2.25
Galaxy S9	1	0	1	0
Galaxy S10	0	0	0	0

Table 3: Average Push Notification Time by Phone Type

blindly forward copies of packets to any available nodes without a selection criteria [1].

Packets for an Android wear service may not deliver directly but rather push the packet to whichever service it comes in contact with first. As a result, some packets may not reach the Android wear service immediately for push determination (to push or not to push) and ultimately can cause a delay in push notifications. In our work, eight of the thirty-two devices displayed delayed behavior in their push notifications when they were removed from the Wi-fi connection and tested under a carrier service.

5.2 Older Version of Phone

Although older phones did not have significant latency in their notification pushes, their performance was slower than newer devices. While newer devices pushed notifications performance immediately, the performance of older devices varied between 5-10 seconds. Also in newer devices, the data storage capability is significantly larger than in older devices. Lower data storage impacts the speed of push notifications, causing older devices or other resource constrained devices to have reduced performance for push notifications, as observed in Section 4.1. This may be a danger for resource constrained devices such as health monitors.

Table 3 provides with a top-level view of the data for all tested smartphones. Based on the analysis, we determine that the push delay is negligible on the newer version of Android phones. The older model devices have small delay behavior in their push notifications. Since the delay time was less then 10 seconds, we considered the behavior as "slower performance" rather than an actual push notification delay. Other devices though such as the Samsung Galaxy S6 in Configuration 3 had a recorded notification delay as the notification push took 15 seconds on average to succeed. While this performance reduction could have occurred due to the carrier service, the age of the device could also be a factor.

5.3 Do Not Disturb Setting

The "do not disturb" setting in the phone provides great benefits to users as it can be set manually for a specific time or automatic start and ending time for every day. Even when the do not disturb setting is enabled, it will allow

bypass calls or messages to the phone if the sender's contact is saved to the users' favorite list. However, when the Android wear service looks for when to push, if the phone is set to the "do not disturb" setting, it will not push any notifications to its connected smartwatch regardless of favorite contact selection. This should be of concern to developers working with safety critical applications.

5.4 Phone Carrier Service

It was expected during our work that the probability of the phone carrier service having an affect on push notification are very low since the push notifications communicate via BT connectivity once the Android service determines to push. Collected data suggested differently. Eight of the thirty-two devices displayed latency when they were tested under their cellular network.

5.5 Applications Updates

Lastly, device performance can be reduced or interrupted by lack of application updates. Even with applications such as an image gallery that may not be directly related to call or text, the overall performance of the device and notifications may degrade. With extra data communication overhead, updates may delay the Android wear service decision making process. As a result, it can delay the push notifications.

6 Threats to Validity

Our study includes push performance data from more than 500 calls in four different configurations. However, it was limited to Android phones along with smartwatches due to accessibility of physical devices. We expect that similar trends will occur with other types of devices as hardware design and architecture had negligible impact on delays. Also, all measurements were performed manually, which threatens the reliability of the timing measurements. To mitigate this risk, each configuration setup on the pairs of devices were run three times, and the average was reported. Finally, the number of factors considered in the NPP analysis and Vector Classification are limited due to only having an outside look of the system at the time of notification. Other factors may contribute to delays besides those considered here.

An automated testing technique will help us to more accurately measure push notification performance delays across a larger number of devices and with a higher level of precision and efficiency in the future. Automation will also support the analysis of a larger set of factors such as hardware state, network state, and more at the time of notification.

7 Related Work

Work exists in the study of push notification processes and DTN architectures for smartphone and wearable devices. Flash memory, which can additionally

influence performance, has also been studied. Tools have also been developed to inspect application behavior and device state. However, to our knowledge, notification push performance has not been analyzed within the environment.

Push notifications are a very important application embedded in smartwatches, which allow paired smartphones to communicate with its connected wearable device and send notifications. Liu et al. [9] identifies that 200+ apps utilize push notifications and exhibit bursty arrival patterns. The push notifications are mostly dominated by instant messaging and email notifications. They identify that 84% of the time the smartwatches were paired with their connected smartphones while the network traffic flows were short, small, and slow. During that time BT traffic was noted to be 91% of the overall traffic.

Most smartphones are designed to receive push notifications as soon as they are available to minimize latency. However, Liu et al. [10] identifies that when a notification is available, phone-side user apps push to the Android wear service policy to determine the notifications push. Many Android apps push notifications to the smartwatch only when other phone-sides apps are not running. When other apps are running on the phone, the service policy assumes that the user is interacting with their device and is able to see the notifications, and therefore, no push is necessary. This service policy is not ideal and can cause unnecessary latency in the push notifications process or even skip notifications. This is because, in some cases, a user may leave an app running on their phone but not be in close proximity to see an incoming call or message.

Extensive use of push notification creates "interruption overload" in mobile devices, and in some cases, it even creates a bottleneck. This interruption occurs when various types of notification pushes take place. For example, there might be an OS update, incoming messages from other users, news alerts, user preferred application alerts, etc.... The notification push is designed to provide the user with alerts in a timely and "instant manner" once a new message or alert is generated by or by performing a periodical pulling. However, as more applications are utilized by users and other services run on the Android devices in the background, incoming notifications are often interrupted and may arrive at a delayed interval instead of instantly [15].

Energy savings and increased battery life play a big role in development of new capability and functionality in connected wearable devices. Because of this, most notifications pushes are used to communicate with delay-tolerant in wearable devices [9], [22]. Delay-tolerant addresses the issue of lack of continuous network connectivity [3]. Benhamida et al. [3] conduct a survey to look into the use of DTN in IoT applications to overcome connectivity problems. The goal is to identify current solutions that enable delay tolerant IoT. In the survey, they identify two main characteristics for DTN that effect IoT scalability: delivery latency and network coverage. Delivery latency measures the duration between a message being generated and its delivery. It also captures efficiency of the routine path [3]. In another publication, Li et al. discuss DTN performance

relays, and they identify that the routine and forwarding algorithms and their design compatibility with mobility patterns are key factors in performance [8]. The Mobile Delay Tolerant Network (MDTN) is used to establish communication when there is a lack of infrastructure. To improve the data accessibility and efficiency, the users tend to utilize cooperative caching schemes, however, in the MDTN, contact patterns remain a challenge [18].

Flash memory is another major component in smart phones that can affect system and notification performance [11]. In a research study by Zao et al., they recognize that the physical characteristics of flash memory and its limited life cycle tend to degrade smartphone's performance over time. Thus, they designed a content-aware trace collection tool with the purpose to examine the data redundancy characteristics in Android phones. During their research study they collected 15 mobile trace for analysis and determined that 20% to 40% of the I/O requests on the I/O critical path of the storage stack are redundant and that this data redundancy is minimally shared among different applications [11]. Because of this, there is a longer response time for applications and a reduced storage capacity in Android smartphones. As observed in our study, application response time and storage capacity have the potential to influence notification push performance.

Mobile application behavior can be observed through a number of tools. For example, hardware performance can be tracked through applications that monitor the CPU, OS, or system [5], [7], [6]. Software state can also be monitored to a limited degree, although the majority of testing is performed within an emulator [21]. To support debugging and replay of application actions on the device under consideration, Sahin et al. [17] developed a tool called RandR. RandR captures and replays multiple sources of input during application execution without need of the application's source code. While their work focused on capturing UI interactions, it could be utilized for other application state monitoring and recording such as when a push notification occurs.

Byrd et. al. [4] took a closer look at these delay behaviors and the Android hardware process during a notification push from a cloud service or other application. They described and developed a framework that focuses on automated hardware profiling and capture of mobile application states. From this framework, they are able to better understand application behavior at the hardware level during a notification push. As part of google cloud service, a Firebase server can log and record various application data including user interaction. When deploying an application, majority of Android devices (models and OS versions) are found to be challenging since many software implementation relays on the device specification and OS [12].

8 Conclusion

Many factors can contribute to notification push delays, and as wearable devices grow in popularity for general and specialized use, delays become an issue. This is

especially true given that many of these devices are being used to assist in health monitoring and related alerts. Some of these alerts could be time sensitive and require immediate attention. Given the similar technologies being used between smartphones and some medical alert devices, the delaying vectors can affect both sets of devices.

In our work, we identified that most newer smartphones performed efficiently, but older devices exhibited delays in their notification pushes. Phone stored data volume and their carrier service also seemed to play a small role in notifications push performance and resulting delay. These vectors might not seem significant, however, it can easily affect a medical wearable device and prevent the alert notification push from performing in a timely manner. Our overall research highlights some of the vectors in the push notification process. More study and research in this area will provide the developers with "lookout factors" when developing wearable medical alert devices.

In future work, we will examine a wider range of factors that may contribute to notification delays across a broader set of devices. We will also automate the testing process focusing on particular applications.

References

1. Abdelkader, T., Naik, K., Nayak, A., Goel, N., Srivastava, V.: A performance comparison of delay-tolerant network routing protocols. IEEE Network **30**(2), 46–53 (2016). DOI 10.1109/MNET.2016.7437024
2. Al-Sharrah, M., Salman, A., Ahmad, I.: Watch your smartwatch. In: 2018 International Conference on Computing Sciences and Engineering (ICCSE), pp. 1–5. Kuwait City, Kuwait (2018). DOI 10.1109/ICCSE1.2018.8374228
3. Benhamida, F.Z., Bouabdellah, A., Challal, Y.: Using delay tolerant network for the Internet of Things: Opportunities and challenges. In: 2017 8th International Conference on Information and Communication Systems (ICICS), pp. 252–257 (2017). DOI 10.1109/IACS.2017.7921980
4. Byrd, R.W., Sultana, T., Walcott, K.R.: Ahpcap: A framework for automated hardware profiling and capture of mobile application states. In: 2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 183–188. IEEE (2020)
5. Google: Cpu monitor - temperature, usage, performance. `https://play.google.com/store/apps/details?id=com.glgjing.stark&hl=en_US`
6. Google: Os monitor - apps on google play. `https://play.google.com/store/apps/details?id=com.eolwral.osmonitor&hl=en_US`
7. Google: System monitor - cpu, ram booster, battery saver. `https://play.google.com/store/apps/details?id=com.glgjing.marvel&hl=en_US`
8. Li, Y., Hui, P., Jin, D., Chen, S.: Delay-Tolerant Network Protocol Testing and Evaluation. IEEE Communications Magazine p. 9 (2015)
9. Liu, X., Chen, T., Qian, F., Guo, Z., Lin, F.X., Wang, X., Chen, K.: Characterizing Smartwatch Usage in the Wild. In: Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys '17, pp. 385–398. ACM Press, Niagara Falls, New York, USA (2017). DOI 10.1145/3081333.3081351. URL `http://dl.acm.org/citation.cfm?doid=3081333.3081351`

10. Liu, X., Yao, Y., Qian, F.: Poster: Improve Push Notification on Smartwatches. In: Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys '17, pp. 154–154. ACM Press, Niagara Falls, New York, USA (2017). DOI 10.1145/3081333.3089298. URL http://dl.acm.org/citation.cfm?doid=3081333.3089298
11. Mao, B., Zhou, J., Wu, S., Jiang, H., Chen, X., Yang, W.: Improving flash memory performance and reliability for smartphones with i/o deduplication. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **38**(6), 1017–1027 (2019). DOI 10.1109/TCAD.2018.2834395
12. Maryam, S., Purwono, A., Syahril: Android application development for push notification feature for indonesian space weather service based on google cloud messaging. Journal of Physics: Conference Series **2214**(1), 012,031 (2022). DOI 10.1088/1742-6596/2214/1/012031. URL https://doi.org/10.1088/1742-6596/2214/1/012031
13. Moustafa, N., Slay, J.: The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set. Information Security Journal: A Global Perspective **25**(1-3), 18–31 (2016)
14. Muhammad, G., Rahman, S.M.M., Alelaiwi, A., Alamri, A.: Smart health solution integrating iot and cloud: A case study of voice pathology monitoring. IEEE Communications Magazine **55**(1), 69–73 (2017). DOI 10.1109/MCOM.2017.1600425CM
15. Okoshi, T., Tsubouchi, K., Tokuda, H.: Real-World Product Deployment of Adaptive Push Notification Scheduling on Smartphones. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2792–2800. ACM, Anchorage AK USA (2019). DOI 10.1145/3292500.3330732. URL https://dl.acm.org/doi/10.1145/3292500.3330732
16. Roegiest, A., Tan, L., Lin, J.: Online In-Situ Interleaved Evaluation of Real-Time Push Notification Systems. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17, pp. 415–424. ACM Press, Shinjuku, Tokyo, Japan (2017). DOI 10.1145/3077136.3080808. URL http://dl.acm.org/citation.cfm?doid=3077136.3080808
17. Sahin, O., Aliyeva, A., Mathavan, H., Coskun, A.K., Egele, M.: RANDR: Record and Replay for Android Applications via Targeted Runtime Instrumentation p. 11 (2019)
18. She, J., Bai, X.: Caching strategy in Mobile Delay Tolerant Network. In: 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 497–500 (2016). DOI 10.1109/ICSESS.2016.7883117
19. Siboni, S., Shabtai, A., Tippenhauer, N.O., Lee, J., Elovici, Y.: Advanced Security Testbed Framework for Wearable IoT Devices. ACM Transactions on Internet Technology **16**(4), 1–25 (2016). DOI 10.1145/2981546. URL http://dl.acm.org/citation.cfm?doid=3023158.2981546
20. Turner, L.D., Allen, S.M., Whitaker, R.M.: Push or Delay? Decomposing Smartphone Notification Response Behaviour. In: A.A. Salah, B.J. Kröse, D.J. Cook (eds.) Human Behavior Understanding, vol. 9277, pp. 69–83. Springer International Publishing, Cham (2015). DOI 10.1007/978-3-319-24195-1_6. URL http://link.springer.com/10.1007/978-3-319-24195-1_6
21. Walcott-Justice, K., Mars, J., Soffa, M.L.: Theme: a system for testing by hardware monitoring events. In: Proceedings of the 2012 International Symposium on Software Testing and Analysis, pp. 12–22 (2012)

22. Yang, Y., Cao, G.: Characterizing and optimizing background data transfers on smartwatches. In: 2017 IEEE 25th International Conference on Network Protocols (ICNP), pp. 1–10. IEEE, Toronto, ON (2017). DOI 10.1109/ICNP.2017.8117536. URL http://ieeexplore.ieee.org/document/8117536/
23. Zachariah, T., Klugman, N., Campbell, B., Adkins, J., Jackson, N., Dutta, P.: The internet of things has a gateway problem. In: Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications, HotMobile '15, pp. 27–32. ACM, New York, NY, USA (2015). DOI 10.1145/2699343.2699344. URL http://doi.acm.org/10.1145/2699343.2699344